

Software and Documentation
©Copyright 1982
Vaisar International

HANDY REFERENCE CARD **valFORTH 1.1**

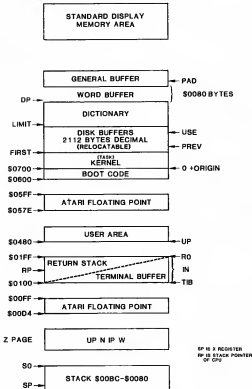
Vocabularies

CONTEXT	[-- addr]	Returns address of pointer to context vocabulary (searched first).
CURRENT	[-- addr]	Returns address of pointer to current vocabulary (where new definitions are put).
FORTH	[--]	Main Forth vocabulary (execution of FORTH sets CONTEXT vocabulary).
EDITPR	[--]	Editor vocabulary; sets CONTEXT.
ASSEMBLER	[--]	Assembler vocabulary; sets CONTEXT.
DISPATCH	[--]	Sets (USER) vocabulary to CONTEXT.
VOCABULARY	[--]	Create new vocabulary named xxx.
XXX	[--]	Print names of all words in CONTEXT vocabulary.
WLIST	[--]	

Miscellaneous and System

!	[--]	Begin comment, terminated by right paren on same line; space after !.
FORGET xxx	[--]	Forget all definitions back to and including xxx.
QUIT	[--]	Error termination of operation.
"xxx	[-- addr]	Find the address of xxx in the dictionary; if used in definition, compile address.
HERE	[-- addr]	Return address of next unused byte in the dictionary.
PRD	[-- addr]	Returns address of scratch area (usually 128 bytes beyond HERE).
PR	[-- addr]	System variable containing offset into input buffer. Word, e.g., by WORD.
TOP	[-- addr]	Return address of top stack item.
ALLST	[n --]	Leave a gap of n bytes in the dictionary.
+	[n --]	Compile a number into the dictionary.

valFORTH Memory Map



Atari is a trademark of Atari, Inc. a division of Warner Communications.

HANDY REFERENCE CARD **valFORTH 1.1**

valFORTH 6502 Assembler

ASSEMBLER (---)	Calls up the assembler vocabulary for subsequent assembly language programming.
CODE xxx (---)	Enters the new word "xxx" into the dictionary as machine language word and calls up the assembler vocabulary for subsequent assembly language programming.
C (---)	Emphasizes an assembly language definition by performing a security check and setting the CONTEXT vocabulary to the same as the CURRENT vocabulary.
END-CODE (---)	A commonly used synonym for the word C; above. The word C; is recommended over END-CODE.
SUBROUTINE xxx (---)	Enters the new word "xxx" into the dictionary as machine language subroutine and calls up the assembler vocabulary for subsequent assembly language programming.
CODE (---)	When the assembler is loaded, puts the system into the assembler vocabulary for subsequent assembly language programming. See main glossary for further explanation.

Control Structures

IF, (flag --- addr 2)	Begins a machine language control structure based on the 6502 status flag on top of the stack. Leaves to address and a security check value for the ELSE or ENDOIF clauses below. "flag" can be EQ, NE, CC, CS, VC, VS, NF, or PL. (Command forms) ...flag...IF...if-true...ENDIF...all... ...flag...IF...if-false...ENDIF...all...
ELSE, (addr 2 --- addr 3)	Used in an IF clause to allow for execution of code only if IF clause is false. If the IF clause is true, this code is bypassed.
ENDIF, (addr 2/3 ---)	Used to terminate an IF control structure clause. Additionally, ENDOIF resolves all forward references. See IF, above for command form.
BEGIN, (--- addr 1)	Begins machine language control structures of the following forms: ...BEGIN...AGAIN... ...BEGIN...flag...UNTIL... ...BEGIN...flag...while-true...REPEAT... where "flag" is one of the 6502 statuses: EQ, NE, CC, CS, VC, VS, NF, and PL.
UNTIL, (addr 1 flag ---)	Used to terminate a post-testing BEGIN clause thus allowing for conditional looping of a program segment while "flag" is false.
WHILE, (addr 1 flag --- addr 4)	Used to begin a pre-testing BEGIN clause thus allowing for conditional looping of a program segment while "flag" is true.
REPEAT, (addr 4 ---)	Used to terminate a pre-testing BEGIN...WHILE clause. Additionally, REPEAT resolves all forward addresses of the current WHILE clause.
AGAIN, (addr 1 ---)	Used to terminate an unconditional BEGIN clause. Execution cannot exit this loop unless a JMP instruction is used.

Parameter Passing (These routines must be jumped to.)

NEXT (--- addr)	Transfers control to the next FORTH word to be executed. The parameter stack is left unchanged.
PUSH (--- addr)	Pushes a 16 bit value to the parameter stack whose low byte is found on the 6502 return stack and whose high byte is found in the accumulator.
PUSHA (--- addr)	Pushes a 32 bit value to the parameter stack whose low byte is found in the accumulator and whose high byte is zero.
PUT (--- addr)	Replaces the value currently on top of the parameter stack with the 16 bit value whose low byte is found on the 6502 stack and whose high byte is in the accumulator.
PUTA (--- addr)	Replaces the value currently on top of the parameter stack with the 32 bit value whose low byte is in the accumulator and whose high byte is set to zero.
BINARY (--- addr)	Smooths the top value of the parameter stack and then performs a PUT operation described above.
POP and POPA (--- addr)	POP drops one value from the parameter stack. POPA drops the values from the parameter stack.
SETUP (--- addr)	Moves one to four values to the N scratch area in the zero page and drops all values held from the parameter stack.
N (--- addr)	Prints to a nine-byte scratch area in the zero page beginning at N-1 and going to N+7.
Decodes (various --- various)	ADC, AND, ASL, BIT, BSR, CLC, CLD, CLI, CLP, CMP, CPY, CPZ, DEC, DEX, DFP, EOR, INC, INX, INY, JSR, JMP, LDA, LDR, LDN, LSR, NOP, ORA, PHB, PHA, PLA, PLP, ROL, ROR, RTI, RTS, SEC, SED, SEI, STX, STY, STZ, TAX, TAY, TXS, TXN, TYS, TYA, TYX, TYZ.

Aliases

NXT, = NEXT JMP,	POPA, = POPA JMP,
PSH, = PUSH JMP,	XL, = ISAVE LDA,
PTH, = PUT JMP,	XS, = ISAVE STX,
PUSHA, = PUSHA JMP,	THEN, = ENDOIF,
PUTA, = PUTA JMP,	END, = UNTIL,
POP, = POP JMP,	

HANDY REFERENCE CARD **valFORTH** **SOFTWARE SYSTEM** **EDITOR 1.1 COMMAND SUMMARY**

Below is a quick reference list of all the commands which the video editor recognizes.

Entering the Edit Mode (executed outside of the edit mode)

- `V` { `scr# ---` }
 - Enter the edit mode and view the specified screen.
- `I` { `---` }
 - In-view the current screen.
- `WHERE` { `---` }
 - Enter the edit mode and position the cursor over the word that caused a compilation error.
- `LOCATE` `cccc` { `---` }
 - Enter the edit mode and position the cursor over the word "cccc" where it is defined.
- `LOCATOR` { `ON/OFF ---` }
 - When ON, allows all words compiled until the next OFF to be locatable using the LOCATE command above.
- `>LINES` { `>lines --` }
 - Sets the length (in lines) of the storage buffer. The default is five.

Cursor Movement (Issued within the edit mode)

- `ctrl` ↑
 - Move cursor up one line, wrapping to the bottom line if moved off the top.
- `ctrl` ↓
 - Move cursor down one line, wrapping to the top line if moved off the bottom.
- `ctrl` ←
 - Move cursor left one character, wrapping to the right edge if moved off the left.
- `ctrl` →
 - Move cursor right one character, wrapping to the left edge if moved off the right.
- `RETURN`
 - Position the cursor at the beginning of the next line.
- `TAB`
 - Advance to next tabular column.

Editing Commands (Issued within the edit mode)

- `ctrl` `INS`
 - Insert one blank at cursor location, losing the last character on the line.
- `ctrl` `DEL`
 - Delete character under cursor, closing the line.
- `shift` `INS`
 - Insert blank line above current line, losing the last line on the screen.
- `shift` `DEL`
 - Delete current cursor line, closing the screen.
- `ctrl` `I`
 - Toggle insert-mode/replace-mode. (See full description of `ctrl-I`.)
- `BACKS`
 - Delete last character typed, if on the same line as the cursor.
- `ctrl` `H`
 - Erase to end of line (back).

Buffer Management (Issued within the edit mode)

- `ctrl` `T`
 - Delete current cursor line sending it to the edit buffer for later use.
- `ctrl` `F`
 - Take the current buffer line and insert it above the current cursor line.
- `ctrl` `X`
 - Copy current cursor line sending it to the edit buffer for later use.
- `ctrl` `U`
 - Take the current* buffer line and copy it to the current cursor line.
- `ctrl` `B`
 - Roll the buffer making the topmost buffer line current.
- `ctrl` `D`
 - Roll the buffer backwards making the fourth buffer line on the screen current.
- `ctrl` `C`
 - Clear the current* buffer line and performs a `ctrl-B`.

*Note: The current buffer line is bottommost on the video display.

Changing Screens (Issued within the edit mode)

- `ctrl` `P`
 - Display the previous screen saving all changes made to the current screen.
- `ctrl` `N`
 - Display the next screen saving all changes made to the current screen.
- `ctrl` `S`
 - Save the changes made to the current screen and end the edit session.
- `ctrl` `Q`
 - Quit the edit session forgetting all changes made to the current screen.

Special Keys (Issued within the edit mode)

- `ESC`
 - Do not interpret the next key typed as any of the commands above. Send it directly to the screen instead.
- `ctrl` `A`
 - Put the arrow "→" ("next screen") in the lower-right-hand corner of the screen unless it is already there, in which case remove it.
- `ctrl` `J`
 - Split the current line into two lines at the point where the cursor is.
- `ctrl` `O`
 - Corrects any major editing blunders.

Screen Management (executed outside of the edit mode)

- `FLUSH` { `--` }
 - Save any updated FORTH screens to disk.
- `EMPTY` { `--` }
 - Forget any changes made to any screens not yet flushed to disk.
- `BUFFERS` { `from to --` }
 - Copies screen `from` to screen `to`.
- `COPY` { `from to --` }
 - Blank fills specified screen.
- `CLEAR` { `scr# --` }
 - Blank fills the specified number of screens starting with screen `scr#`.
- `CLEAR` { `scr# screens --` }
 - Blank fills the specified number of screens starting with screen number "from".
- `SMOKE` { `from to screens --` }
 - Duplicate the specified number of screens starting with screen number "from".